INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Duplicate address detection and autoconfiguration in OLSR

Saadi Boudjit, Anis Laouiti, Paul Muhlethaler and Cédric Adjih

No 5475

Janvier 2005

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--5475--FR+ENG

Duplicate address detection and autoconfiguration in OLSR

Saadi Boudjit, Anis Laouiti, Paul Muhlethaler and Cédric Adjih

Thème 1 — Réseaux et systèmes
Projet HIPERCOM

Rapport de recherche n ° 5475 — Janvier 2005 — 20 pages

Abstract: Mobile Ad hoc NETworks (MANETs) are infrastructure-free, highly dynamic wireless networks, where central administration or configuration by the user is very difficult. In hardwired networks nodes usually rely on a centralized server and use a dynamic host configuration protocol, like DHCP [9], to acquire an IP address. Such a solution cannot be deployed in MANETs due to the unavailability of any centralized DHCP server. For small scale MANETs, it may be possible to allocate free IP addresses manually. However, the procedure becomes impractical for a large-scale or open system where mobile nodes are free to join and leave. Most of the autoconfiguration algorithms proposed for ad hoc networks are independent of the routing protocols and therefore, generate a significant overhead. Using the genuine optimization of the underlying routing protocol can significantly reduce the autoconfiguration overhead. One of the MANET protocols which have been recently promoted to RFC is the *OLSR* routing protocol [13], on which this article focuses. This article aims at complementing the *OLSR* routing protocol specifications to handle autoconfiguration. The corner stone of this autoconfiguration protocol is an advanced duplicate address detection algorithm. Under well defined assumptions, we prove the correctness of the the proposed autoconfiguration protocol.

Key-words: MANET (Mobile Ad hoc NETworks); Autoconfiguration; OLSR

(Résumé : tsvp)

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : 01 39 63 55 11 - International : +33 1 39 63 55 11
Télécopie : (33) 01 39 63 53 30 - International : +33 1 39 63 53 30

Détection d'adresses dupliquées et autoconfiguration pour OLSR

Résumé : Les réseaux mobiles ad hoc sont des réseaux sans infrastructure et hautement dynamiques. Dans ces réseaux, il est très difficile de prévoir une administration centralisée ou d'utiliser une configuration manuelle du réseau. Les noeuds des réseaux filaires utilisent un serveur centralisé et le protocole DHCP pour acquérir une adresse IP. Une telle solution ne peut être utilisée dans les réseaux mobiles ad hoc car il n'existe aucun serveur DCHP centralisé. Pour de petits réseaux, il reste possible d'allouer les adresses disponibles à la main. Cependant une telle procédure devient impossible pour de grands réseaux ou dans des réseaux de mobiles ouverts où les mobiles peuvent rejoindre ou quitter le réseau librement. La plupart des algorithmes d'autoconfiguration proposés pour des réseaux ad hoc sont indépendant du protocole de routage utilisé; ils génèrent ainsi un "overhead" important. L'utilisation des optimasations propres du protocole de routage peut réduire significativemnet ces derniers. Un des protocoles de routage qui a été promu récemment RFC est le protocole de routage *OLSR* [13] autour duquel cet article est. Cet article cherches à compléter les spécifications du protocole pour traiter l'autoconfiguration. La pierre angulaire de ce protocole d'autoconfiguration est un mécanisme évolué de détection d'adresses dupliquées. Sous des hypothèses bien définies nous prouvons le bon fonctionnement du protocole d'autoconfiguration proposée.

Mots-clé : Réseau mobile ad hoc, Autoconfiguration, OLSR

1 Introduction

Many fruitful efforts have focused on routing protocols for *MANET* in recent years, dealing essentially with the routing issue itself, and not necessarily considering the autoconfiguration functionality. These MANET protocols can be classified into proactive protocols [13] where each node maintains an up-to-date version of the network topology by periodic exchange of control messages with neighboring nodes; and reactive protocols [14] where each node discovers the route to a destination on demand.

Research on automatic configuration of IP addresses for *MANET* is relatively less frequent. The IPv6 and ZEROCONF working groups of the IETF deal with autoconfiguration issues but with a focus on wired networks. Automatic address allocation is more difficult in a *MANET* environment than in wired networks due to instability of links, mobility of the nodes, the open nature of the mobile ad hoc networks, and lack of central administration in the general case. Thus performing a DAD (Duplicate Address Detection) generates more complexity and more overhead in ad hoc networks than in wired networks where protocols such as DHCP [9] and SAA [7] can be used.

In this research report we will describe an autoconfiguration solution for the *OLSR* protocol. This solution is based on an efficient Duplicate Address Detection (DAD) algorithm which takes advantage of the genuine optimization of the *OLSR* protocol.

The research report is structured as follows: section 2 is dedicated to related work, mostly concerning previously proposed autoconfiguration protocols in ad hoc networks. Then a summary of the *OLSR* protocol, with its main features, is described in section 3. Section 4 describes the duplicate address detection mechanism which is the core of our proposed autoconfiguration protocol. A formal proof of correctness of this detection algorithm is given. Section 5 proposes different ways to assign an initial address to a newly arriving node and to resolve address duplication. This section actually completes the previous one on duplicate address detection. These two sections constitute a complete autoconfiguration algorithm. The research report concludes in section 6.

2 Related work

Numerous studies have been carried out on autoconfiguration protocols and the related issue of duplicate address detection in ad hoc networks. These studies have been proposed within the IETF or published in academic papers. This section is organized as follows. First, we review autoconfiguration scenarios and the different conditions where address duplications may occur. Then, we briefly present the various proposed autoconfiguration protocols. To conclude this section we position our contribution with respect to the previously proposed autoconfiguration protocols.

2.1 Address autoconfiguration scenarios and address duplications

The first scenario is the simplest: a mobile node joins and then leaves a *MANET*. An unused IP address is allocated to the node on its arrival and becomes free on its departure. In some scenarios, the allocated IP address may be duplicated in the network.

A more complicated scenario is the following: nodes are free to move arbitrarily in the *MANET* and, consequently, the network may becomes partitioned. In the resulting partitions, the nodes continue to use the previously allocated IP addresses. If a new node comes to one of the partitions, it may be assigned an IP address belonging to another partition. Address duplication may occur when the partitions merge.

Another scenario is when two independent *MANETs* merge. Because the two *MANETs* were configured separately, and the address allocation in each of the *MANETs* is independent of the other, there may be duplicated addresses when the two *MANETs* merge.

Most of the other scenarios, can be thought of as special cases of the three scenarios described above.

2.2 Address autoconfiguration in ad hoc networks

There are two main approaches to address autoconfiguration in ad hoc networks. The first approach tends to allocate conflict free addresses. The algorithms in this approach are often called conflict-free allocation algorithms. The second approach tends to allocate addresses on a random basis and uses dedicated mechanisms to detect duplicate addresses. When duplications are detected, new addresses with new values are assigned.

The Distributed Dynamic Host Configuration Protocol (DDHCP) [22] is one example of a conflict-free allocation algorithm. In this algorithm, the nodes responsible for allocation try to assign an unused IP address to a new node – unused to the best of their knowledge. Then the new node performs DAD to guarantee that it is an unallocated IP address. DDHCP maintains a global allocation state, so IP addresses which have been used, and addresses which have not yet been allocated, are known. When a new node joins the network, one of its neighbors choses an unused address for it. The same unused IP address in the global address pool could be assigned to more than one node arriving at almost the same time. This is the reason why DAD is still performed by a node after getting an IP address. This algorithm takes into account network partition and merger, and works well with proactive routing protocols.

Dynamic Configuration and Distribution Protocol (DCDP) [23] is another conflict-free allocation algorithm. When a new mobile node joins the *MANET*, an address pool is divided into halves between itself and a configured node. This algorithm takes into account network partition and merge, however conflicts will occur during the merge if two or more of the separately configured *MANETs* taking part in the merge, begin with the same reserved address range.

Another conflict-free allocation algorithm, called the 'prophet allocation protocol' has been proposed for large scale *MANETs* in [25]. The idea is that every mobile node executes a stateful function *f(n)* to get a unique IP address. *f(n)* is function of a state value called

the *seed* which is updated for each node in the network. In this algorithm, mechanisms are proposed for network partition and merger. The difficulty in this solution is to find a function *f(n)* which will guarantee the generation of unique IP addresses each time the function is executed by a node.

The algorithms related to the second approach, perform a DAD (Duplicate Address Detection) to ensure the uniqueness of the allocated IP address. The general procedure is that a node generates a tentative address and then performs DAD within its neighborhood (radio range of the node). If the address is unique, the DAD is performed again over the whole network and a unique IP address is constructed. Examples of such approaches include [1], [17] and [18]. DAD mechanisms can also be divided into two categories which differ in when, and how duplicate addresses are detected.

The ADAD (Active Duplicate Address Detection) mechanisms distribute additional information in the network to prevent address duplication as, for instance, in [17] and [18].

In contrast, PDAD (Passive Duplicate Address Detection) algorithms ([19] and [20]), try to detect duplicates without disseminating additional control information in the network. The idea behind this approach is to continuously monitor routing protocol traffic to detect duplicates rather than sending additional control packets for this purpose. However, in ([20]) a so-called Address Conflict Notification (ACN) message is introduced for the purpose of conflict resolution, and no less than nine different approaches to detecting duplicated addresses have been presented for proactive link-state routing protocols. These approaches are based on: the *Sequence Numbers(PDAD-SN)*, the *Sequence Numbers Difference(PDAD-SND)*, the *Sequence Numbers Equal(PDAD-SNE)*, the *Neighborhood History(PDAD-NH)*, the *Link State algorithm(PDAD-LS)*, the *Locality Principle (PDAD-LP)*, the *Sequence Number Increment(PDAD-SNI)*, the *Source Address algorithm(PDAD-SA)*, and the *Duplicate Cache algorithm(PDAD-DC)*. Three algorithms have been proposed for on-demand protocols: The *RREQ-Never-Sent algorithm (PDAD-RNS)*, the *RREP-without-RREQ algorithm(PDAD-RwR)*, and the *2RREPs-on-RREQ algorithm(PDAD-2RoR)*. Each of these approaches is dedicated to a special case and a special scenario, and for their applicability to known routing protocols such as *OLSR* and *AODV*, the author recommends a combination of a subset of the proposed algorithms for each routing protocol.
The advantage of this solution consists in introducing only one additional message used to report a conflict. This saves bandwidth in the absence of conflictual situations. However several drawbacks can be found. First it is a heavy solution because of the numerous algorithms that must be combined to handle all the scenarios. For instance for OLSR, due to the peculiar difficulties induced by the MPR optimization, more than nine different algorithms are necessary to cover all the identified scenarios. At the same time to avoid possible storm effects when a duplicate address is detected, special techniques must be used to control the flooding of ACN messages. Finally there is no formal proof that all duplications can be finally detected.

2.3 Our contribution

In this research report we are proposing a new autoconfiguration algorithm for OLSR based on a DAD approach. The main idea is that preventing address collision requires an assumption that some unique identifiers (*Node-ID*) are allocated to each node in the network. As for the PACMAN proposal, this new autoconfiguration algorithm is optimized to reduce the bandwidth utilization. As a matter of fact this approach uses the OLSR's MPR optimization to broadcast a new control packet(*MAD: Multiple Address Decalaration*) used by the DAD procedure. Moreover this approach is extremely simple and a formal proof of correctness is given in the article. We are now going to give a short review of the *OLSR* routing protocol.

3 OLSR

This section describes the main features of the *OLSR* (Optimized Link State Routing) protocol.

OLSR is an optimization of a pure link state routing protocol. It is based on the concept of *multipoint relays* (*MPRs*) [16]. First, using *multipoint relays* reduces the size of the control messages: rather than declaring all links, a node declares only the set of links with its neighbors that are its "*multipoint relays*". The use of *MPRs* also minimizes flooding of control traffic. Indeed only *multipoint relays* forward control messages. This technique significantly reduces the number of retransmissions of broadcast control messages [13, 16]. The two main *OLSR* functionalities, Neighbor Discovery and Topology Dissemination, are now detailed. Then we present *OLSR* "gateway" mechanism used by some nodes to declare reachability to their connected hosts and networks.

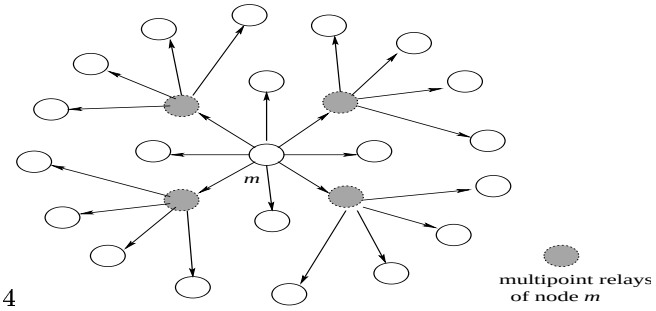
-5pt

3.1 Neighbor Discovery

Each node must detect the neighbor nodes with which it has a direct link.

For this, each node periodically broadcasts *Hello* messages, containing the list of neighbors known to the node and their link status. The link status can be either *symmetric* (if communication is possible in both directions), *asymmetric* (if communication is only possible in one direction), *multipoint relay* (if the link is symmetric and the sender of the *Hello* message has selected this node as a *multipoint relay*), or *lost* (if the link has been lost). The *Hello* messages are received by all 1-hop neighbors, but are not forwarded. They are broadcasted once per refreshing period called the "*HELLO_INTERVAL*". Thus, *Hello* messages enable each node to discover its 1-hop neighbors, as well as its 2-hop neighbors. This neighborhood and 2-hop neighborhood information has an associated holding time, the - "*NEIGHBOR_HOLD_TIME*", after which it is no longer valid.

On the basis of this information, each node independently selects its own set of *multipoint relays* among its 1-hop neighbors in such a way all 2-hop neighbors of m have *symmetric*



4

Figure 1: Multipoint relays of node m

links with $MPR(m)$. This means that the *multipoint relays* cover (in terms of radio range) all 2-hop neighbors (Figure 1) One possible algorithm for selecting these *MPRs* is described in [16]. The *multipoint relay* set is computed whenever a change in the 1-hop or 2-hop neighborhood is detected. In addition, each node m maintains its "*MPR selector set*". This set contains the nodes which have selected m as a *multipoint relay*. Node m only forwards broadcast messages received from one of its *MPR selectors*.

3.2 Topology Dissemination

Each node of the network maintains topological information about the network obtained by means of *TC (Topology control)* messages. Each node m selected as a *multipoint relay*, broadcasts a *TC* message at least every "*TC_INTERVAL*". The *TC* message originated from node m declares the *MPR selectors* of m. If a change occurs in the *MPR selector* set, the next *TC* can be sent earlier. The *TC* messages are flooded to all nodes in the network and take advantage of *MPRs* to reduce the number of retransmissions. Thus, a node is reachable either directly or via its *MPRs*. This topological information collected in each node has an associated holding time "*TOP_HOLD_TIME*", after which it is no longer valid.

The neighbor information and the topology information are refreshed periodically, and they enable each node to compute the routes to all known destinations. These routes are computed with Dijkstra's shortest path algorithm [26]. Hence, they are optimal as concerns the number of hops. The routing table is computed whenever there is a change in neighborhood or topology information.

3.3 OLSR "gateways"

Each node maintains information concerning which nodes may act as "gateways" to associated hosts and networks. These "gateways" periodically generate a *HNA (Host and Network Association)* message, containing pairs of (network address, netmask) corresponding to the

connected hosts and networks. The *HNA* messages are flooded to all the nodes in the network by the *MPRs*. These messages should be transmitted periodically every *HNA_INTERVAL*. The collected information is valid for *HNA_HOLD_TIME*. The networks and associated hosts are added to the routing table and they have the same next hop as the one to reach the appropriate "gateway".

3.4 Multiple Interface Declaration

In the full OLSR protocol, a node which has several interfaces, periodically emits a special type message, "Multiple Interface Declaration", in which it lists all its interfaces addresses, along with one of them, which is fixed, and is (arbitrarily) chosen as its main address.

4 Duplicate Address Detection

Our proposed autoconfiguration algorithm is based on two steps. In the first step, an IP address is selected by the arriving node and this latter can join the ad hoc network. Numerous schemes can be used to select this IP address. For instance the node can perform a random selection in a well known pool of addresses; another technique consists of one of its neighbors selecting the address on behalf of the arriving node. In section 5 we discuss in detail various ways to assign an IP address to an arriving node.

After this first step has been performed, the second step can take place. The aim of this step is to detect potential address duplications on run. To perform this task a DAD algorithm is started on this newly configured node. This DAD algorithm allows the newly configured node to state whether the selected address is duplicated or not in a proactive manner. If such a case occurs, a node can change its address with respect to some specified criteria.

In order to detect address conflicts, each node diffuses a special message that we call a MAD for "Multiple Address Declaration" to the entire network. This control packet includes the node address and the node identifier. The node identifier is a sequence of bits of fixed length L which is randomly generated. Hence we are using the standard idea that the probability of two nodes having the same node identifier is low, and the probability of at least one address collision with N nodes, which is the well known "birthday problem", can be set arbitrarily low by choosing a large enough value of L.

A node detects an address conflict when it receives a MAD message having the same address as its own, but with a different identifier. On the other hand and as shown in figure 2, other nodes will detect the conflict. These nodes could announce the conflict using a special control message as is done in the Pacman protocol [20]. However this approach may induce broadcast storm since many nodes may announce the conflict and special care must be taken to avoid this effect. For that reason we do not recommend this way. An efficient manner to notify the address duplication to the nodes in conflict, consists in allowing the MAD packets to reach all the nodes in the network. To save the channel bandwidth the MAD packets should be broadcasted using the MPR flooding. Actually, applying OLSR

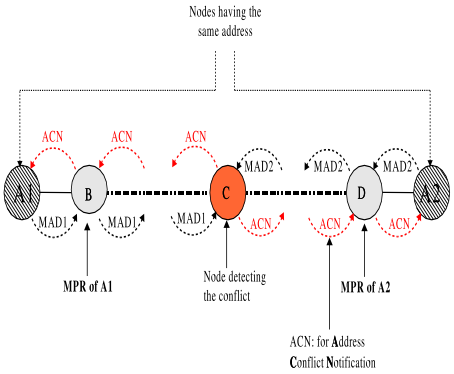


Figure 2: Conflict Notification

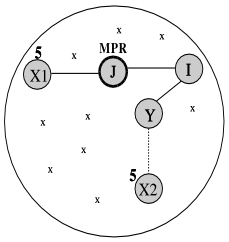


Figure 3: Address duplicate scenario

relaying optimization rules as they are defined, may not be sufficient to ensure diffusion in some conflictual cases. As an illustration of such possible situations, we give the following example.

Figure 3 shows two conflicting nodes $X1$ and $X2$ ($X1$ and $X2$ have the same address 5), in the 2-hop neighbors of node I. In this configuration, nodes Y and I are not chosen as MPRs, then, the "Multiple Address Declaration" messages diffused respectively by the nodes $X1$ and $X2$ can not be propagated throughout the entire network. In our scenario, node I could not calculate its MPR set correctly, because MPR calculation is based on

the assumption that there is no address duplication in the 1-hop and 2-hop neighbors. Consequently, node $X1$ and node $X2$ will not detect the address conflict, and the network remains corrupted. To handle such cases, new rules for MPR flooding algorithm for MAD message diffusion are used. We call it the DAD-MPR Flooding algorithm, and it is detailed in the following section. This modified version of MPR flooding takes into consideration the possibility that MPR originator addresses might be duplicated.

The DAD-MPR flooding algorithm is only useful for "Multiple Address Declaration", but could be used in general for any kind of message which includes the node identifier. This version of MPR flooding is called *Duplicate Address Detecting MPR Flooding* or *DAD-MPR Flooding*

4.1 Special rules of the DAD-MPR flooding and proof of correctness

DAD-MPR flooding should have the property that it will continue to work even in the presence of duplicate addresses in the network. This property means that, in the absence of packet loss, the DAD-MPR Flooding will allow a MAD packet to reach all the nodes in the network. Actually we will not always be able to have exactly this property. However we will have the weaker property stating that for two nodes A_1 and A_2 using the same address A, but holding different node identifiers ID_{A_1} and ID_{A_2} the DAD message of A_1 will be received by A_2 and vice versa the DAD message of A_2 will be received by A_1.

To define and prove the correctness of the DAD-MPR flooding algorithm we will make the following assumptions. We will assume that there are only two nodes with a duplicated address in the network. In the following, we will see that this assumption can actually be significantly weakened.

Let us consider one pair of nodes which have duplicate addresses. Let us denote by A_1 and A_2 these two nodes with same address A, but different node identifiers ID_{A_1} and ID_{A_2} and let us denote by d the distance, in number of hops, between A_1 and A_2. Let us assume that both of them send a MAD message M_1 and M_2.

We intend to add special rules to the MPR flooding algorithm to handle the MAD messages. With these additional rules we must be in a position to prove that if there is no packet loss the MAD message of A_1 will be received by A_2 and vice versa the MAD message of A_2 will be received by A_1. The main problem here comes from the MPR calculation as we have already seen in a previous example. As a matter of fact, the MPR calculation is done on addresses so the MPR set of a node may not be properly calculated if it has address duplication in its 2-hop neighborhood. Therefore, the MPR set may not properly cover the 2-hop neighborhood of this node. In such a case, control messages using the MPR optimization may not be propagated to all nodes in the network.

Let us denote by N_i the set of nodes which are exactly at distance i of A_1 and at distance $d-i$ of A_2, for $i \in \{1, ..., d-1\}$. Those nodes are precisely the nodes which are on a shortest path from A_1 to A_2. Hence the sets N_i are never empty since a shortest path exists. An example of such sets is illustrated in figure 4.

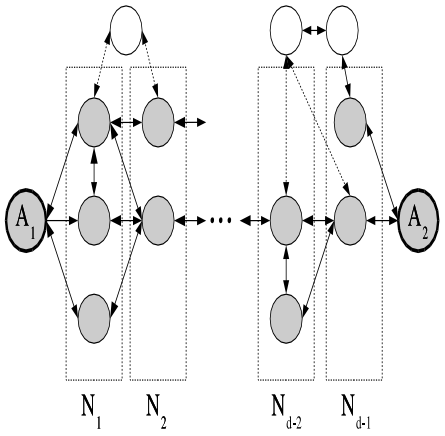


Figure 4: Example of Sets N_i

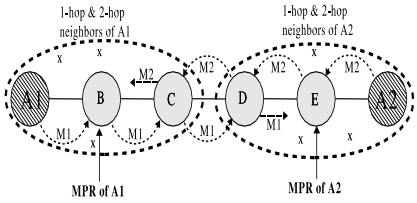


Figure 5: *Distance* ≥ 5

Then several cases can occur, depending on the distance d:

4.1.1 $d \geq 5$

In figure 5 nodes A_1 and A_2 have the same address, and they are 5 hops away from each other. In this case nodes A_1 and A_2 and all the intermediary nodes do not have duplications

in their 1-hop and 2-hop neighbors according to our assumption of only two nodes with a duplicated address. Hence all the intermediary nodes calculate their MPR set properly. So, using the MPR flooding the messages M_1 and M_2 will be correctly propagated to the nodes A_2 and A_1 respectively and the conflict will be detected. With the same consideration we can show that, in this case, all the MAD messages will reach all the network nodes.

4.1.2 $d = 4$

In figure 6 nodes A_1 and A_2 do not have duplications in their 1-hop and 2-hop neighbors according to our initial assumption. Therefore, the MAD messages diffused by A_1 and A_2 will reach node C. Node C can detect the conflict by examining the node identifiers contained in the received MAD messages. These messages should be relayed by C, because it has been chosen as an MPR by B and D. In our case, this is not trivial. In fact, messages generated by A_1 and A_2 may have close sequence numbers, which may prevent one of the two sets of messages from being relayed by C (due to possible existence of a duplicate tuple indicating that such a message having the same sequence number and originator, has been received and processed). We need here to add an extra rule to allow MAD message relaying. We modify the MAD duplicate message detection, which will be based on the node originator address, the message sequence number, plus the node identifier. Hence, A_1 and A_2 MAD messages will be forwarded by C in all cases and reach B and D. Notice here that the MPR calculation of C is affected due to the presence of duplicated addresses in its 2-hop neighbors (A_1 and A_2). C chooses only one node between B and D as an MPR to cover its 2-hop neighbors. The chosen MPR, should act like node C as described before to relay the MAD messages. Following this rule, we are sure that one of the two nodes (A_1 and A_2) receives the MAD messages generated by the other node and can detect the conflict. We will see with the next case, where $d = 3$, another rule that allows the two conflicting nodes to detect the address duplication.

4.1.3 $d = 3$

In case of distance $d = 3$, nodes B and C (figure 7) do not need to choose an MPR to cover respectively their 2-hop neighbors A_2 and A_1 since they have the same address. Address A is considered as a one hop neighbor. In contrast, A_1 chooses node B as a MPR to reach node C, and node A_2 chooses C as a MPR to reach node B. In this situation, and thanks to the new rule described previously, the A_1 MAD messages will reach node C, and the ones generated by A_2 will arrive at node B. But, B and C do not choose each other as an MPR, consequently, A_1 can not receive MAD messages coming from A_2, and A_2 MAD messages will never reach A_1 node. To tackle this problem, we add another rule to enable MAD relaying for such situations, as follows: if a given node N receives a MAD message from a neighbor M, and M did not select N as an MPR, then, node N will repeat this message if it detects that one of its 1-hop neighbors has the same address as the one contained in the MAD message. The MAD TTL value is put to 1 to avoid the transmission of the MAD message beyond the conflicting nodes.

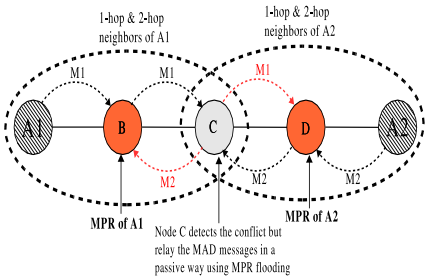


Figure 6: Distance = 4

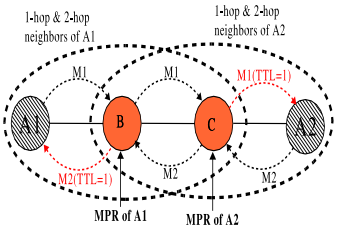


Figure 7: Distance = 3

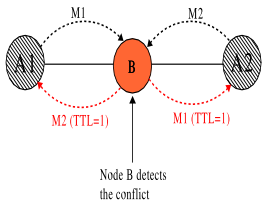


Figure 8: Distance = 2

4.1.4 $d = 2$

In figure 8, the node B detects the duplication because the nodes $A1$ and $A2$ are in its 1-hop neighborhood, it proceeds by the same manner as in the case of $d = 3$. Thus node A_1 will receive the MAD message of A_2; and node A_2 will receive the MAD message of A_1.

4.1.5 $d = 1$

This is the simplest case and because nodes A_1 and A_2 are in the radio range of each other, the conflict will be detected by both nodes.

Actually we can weaken the assumption that the network has only two nodes holding a duplicated address. Let us assume that we have only a duplicate address in two nodes which are four hops away or less from each other. We may have other duplicated addresses but on couples of nodes five hops away or more from each other. In such a case the previous reasoning will still work. In fact the previous analysis uses the assumption that the flooding on a path between two nodes with duplicated addresses will work. Let us now consider other nodes with duplicated addresses. By assumption we are sure that there is no other address duplication for two nodes at a distance less than four. Thus the other address duplications can only occur between nodes at a distance of five or more, and there can not be any 2-hop conflicts due to this other duplication. The MPR flooding will thus work properly; which is precisely the only assumption we have used in the previous reasoning.

In other words, our DAD mechanism will operate correctly under the above assumption. Thus it will operate correctly in the case of an arriving node selecting an IP address but also in the case of partition and merge or even in the case of a direct merge of separately

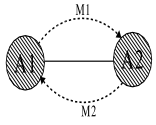


Figure 9: Distance = 1

configured networks. Of course in these last cases the *configuration* of duplicated address will follow the assumption about the distance between nodes holding duplicated addresses i.e only a couple of duplicated addresses with nodes at distance (in terms of hops) of four or less.

Naturally, in these cases the *locations* of duplicated addresses will follow the assumption about the distance between nodes holding duplicated addresses i.e. only a couple of duplicated addresses with nodes at distance (in terms of hops) of four or less can be accepted. Other assumptions can be used. For instance the previous proofs can be obtained with more than a couple of nodes in conflict at a distance of four hops or less. We will just have to assume that the pair of nodes in conflict are far enough apart to avoid interference in their 2-hop neighbor conflicts. Sufficient conditions expressed in terms of distance between nodes in conflict can be easily obtained.

4.2 Specification of the DAD-MPR Flooding algorithm

Let us recall the assumptions here.

Each node A periodically sends a message M including:

- The originator address of A, $Orig_A$, in the OLSR message header.
- The message sequence number, $mssn$, in the OLSR message header.
- The node identifier ID_A (a string of bits) in the message itself.

The message is propagated by MPR flooding to the other nodes ; but for DAD-MPR Flooding, the duplicate table of OLSR is modified, so that it also includes the node identifier list in the duplicate tuple. That is, a duplicate tuple, includes the following information:

- The originator address (as in OLSR standard duplicate table).
- The message sequence number (as in OLSR standard duplicate table).
- The list of node identifiers.

The detailed algorithm for DAD-MPR Flooding is the following:

- When a node receives a message M from node B with originator A, with message sequence number $mssn$, and with node identifier ID_A, it performs the following tasks:
  1. **If** a duplicate tuple exists with the same originator A, the same message sequence number, and ID_A is in the list of node identifiers, **Then**, the message is ignored (it has already been processed). The algorithm stops here.
  2. **Else** one of the following situations occurs :
     (a) A duplicate tuple exists with the same originator A and the same message sequence number, but ID_A is not in the list of node identifiers: then, a conflict is detected (address A is duplicated). ID_A is added to the list of node identifiers.
     (b) A duplicate tuple exists with the same originator A, but with a different message sequence number and ID_A is not in the list of node identifiers: then, a conflict is detected (address A is duplicated). A duplicate tuple is created with the originator address, message sequence number and list of node identifiers containing only ID_A.
     (c) No duplicate tuple exists. A new one is created with the originator address, message sequence number and list of node identifiers containing only ID_A.
  3. The MAD messages should be relayed if one or both of the following rules are met:
     (a) B had chosen this current receiving node as an MPR.
     (b) One of the conflicting nodes is a neighbor of the node detecting the duplication. In such a case, the TTL value of the MAD message showing the conflict is set to one before its retransmission. This also applies even if the current node has not been selected as a MPR by the previous message sender.

5 Address assignment and resolution of conflicts

5.1 Initial address assignment

We have two main ways to allocate an address to a newly arriving node. The first way is to assign this node a random address in the pool of addresses that can be allocated and then to

rely on the DAD algorithm to discover potential conflicts. The second way is to ask for the help of a neighbor node. This neighbor will be able to propose to the newly arriving node a configuration address. Since a neighbor node must in principle receive the MAD messages of all nodes in the network, it can maintain a pool of non-affected addresses.

A newly arriving node can choose between these two approaches taking into account the number of nodes in the network denoted by N_n and the number of addresses in the pool of addresses that have not already been allocated, denoted by N_a. With a random assignation of the addresses, the probability of an address being duplicated can be easily derived as

$$1 - \frac{N_a!}{N_a^{N_n}(N_a - N_n)!}.$$

If we use the Steerling formula we can express an approximation of this probability as $1 - e^{-N_n}(1 - N_n/N_a)^{N_n - N_a - 1/2}$. When N_a is large compared to N_n this probability can be further approximated as

$$1 - (1 - \frac{N_n}{N_a})^{N_n - 1/2}.$$

Let us now assume that a newly arriving node asks one of its neighbors for a new address. We will moreover assume that owing to the MAD messages this neighbor knows almost all the assigned addresses N_n, however due to packet loss, we assume that N_h assigned addresses will not be known by the neighbor. Thus if the neighbor picks an address at random among the addresses that it believes to be available, the probability that a proposed address will be duplicated can be expressed by $1 - \frac{(N_a - N_n + N_h)!}{N_a^{N_h}(N_a - N_n)!}$. Using the Steerling formula this probability can be easily approximated by

$$1 - (eN_a)^{-N_h} \frac{(N_a - N_n + N_h)^{N_a - N_n + N_h + 1/2}}{(N_a - N_n)^{N_a - N_n + 1/2}}.$$

When N_a is large compared to N_n this probability can be further approximated as

$$1 - (1 - \frac{N_n - N_h}{N_a})^{N_h}.$$

5.2 Pool of addresses

The pool of addresses could be for local use only. For example, it could be reserved by the IANA authority for local MANET forwarding (i.e, those addresses must not be forwarded outside the MANET network, nor reached from outside). A second possibility consists in relying on some machines which will announce the prefix to use for address autoconfiguration for this MANET network. These machines could be connected to the internet, and act as gateways. In this case, the addresses may be global addresses, and could be seen from outside.

5.3 Resolution of a conflict

When two nodes A_1 and A_2 are configured with the same IP address and assuming that there is no packet loss, each of these two nodes will receive the MAD message of the other node. Thus the nodes where the conflict lies are bound to discover the conflict. A simple rule to solve this conflict will be: the node in conflict with the smallest identifier changes its address. Since this node knows via the reception of the MAD control messages the already assigned addresses, the new address must be selected at random among the addresses that are believed to be free. To be completely rigorous we can not be sure that all the MAD control messages will be received by all the network nodes. As previously shown, a wrong calculation of the MPR set may affect the broadcast of control messages. The additional rules added to the DAD-MPR flooding algorithm can only deal with the delivery MAD control messages of the conflicting nodes. The MAD control messages of other nodes may be stopped. However in most cases the MAD control messages will be correctly broadcasted. The only reason for non delivery of these MAD control messages is when there is only one route for MAD delivery, and, additionally when there are conflicts in the 2-hop neighborhood of nodes on this route.

Moreover, if a fixed part of the first bits of the identifier is set to the "priority" of the node, this will lower the probability of "important" nodes having to change their addresses.

5.4 Control overhead of the DAD algorithm

The overhead of the proposed autoconfiguration protocol can be easily evaluated. The main part of this overhead comes from the sending of the MAD message. If we assume a network with a maximum of 10000 nodes, eight bytes are enough to code the random identifier. The above formulae (applied with $N_n = 10000$ and $N_a = 2^{48}$) make it possible to compute the collision probability with these figures and a probability of around 10^{-11} is found with 10000 nodes. With this figure, the overhead is 8 bytes for the MAD message itself, 12 bytes for the message header, 4 bytes for the message header and 28 bytes for the UDP IP header. The sum is 52 bytes and if the MAD is sent at the same rate as the TC messages, one every five seconds, the bandwidth used per node is around 80 bits per second.

To give an evaluation of the relative increase in control overhead induced by the MAD messages, we will assume that each node in the network is chosen as an MPR by an average of 6 nodes. The size of the TC is thus $12 * 8 + 32 * 6 + 32 = 320$ bits as the size of an MAD message is $12 * 8 + 64 = 160$ bits. Thus the relative increase in the control overhead is 33%. As a consequence the overhead induced by the MAD control messages remains limited.

6 Conclusion

The autoconfiguration procedure proposed in this article relies on an efficient and proven detection algorithm. A special control message MAD (Multiple Address Declaration) conveys a random identifier with the address of the node. This control message uses the OLSR

genuine MPR flooding algorithm, however special rules have been added to ensure that even with address duplications the MAD will be propagated between the nodes in conflict. A formal proof of correctness of the detection scheme is given in this research report. Simple approaches to allocate an address to a newly arriving node or to solve conflicts are also provided.

References

[1] A. Laouiti, S. Boudjit, P. Minet and C. Adjih, *OLSR for IPv6 networks*, Proceedings of Med-Hoc-Net 2004, June 2004, Bodrum, Turkey.

[2] Charles E. Perkins, Jari T. Malinen, Ryuji Wakikawa, Anders Nilsson and Antti J. Tuominen, *Internet Connectivity for Ad Hoc Networks*, pp 465–482, Wireless Communication and Mobile Computing, number 5, volume 2, August 2002

[3] R. Hinden, S. Deering, "IP Version 6 Addressing Architecture", IETF RFC 2373, July 1998.

[4] R. Hinden, S. Deering, "IPv6 Multicast Address Assignments", IETF RFC 2375, July 1998.

[5] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6)Specification", IETF RFC 2460, December 1998.

[6] T. Narten, E. Nordmark, W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", IETF RFC 2461, December 1998.

[7] S. Thomson, T. Narten, "IPv6 Stateless Address Autoconfiguration", IETF RFC 2462, December 1998.

[8] A. Conta, S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", IETF RFC 2463, December 1998.

[9] R. Droms, Ed.,J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", IETF RFC 3315, July 2003.

[10] R. Hinden, S. Deering, "IP Version 6 Addressing Architecture", IETF RFC 3513, April 2003.

[11] R. Hinden, S. Deering,E. Nordmark, "IPv6 Global Unicast Address Format", IETF RFC 3587, August 2003.

[12] P. Jacquet, P. Muhletaler, A. Qayyum, A. Laouiti, T. Clausen, L. Viennot, "Optimized Link State Routing Protocol" , IEEE INMIC Pakistan, Dec 2001.

[13] P. Jacquet, P. Muhletaler, P.Minet, A. Qayyum, A. Laouiti, T. Clausen, L. Viennot, *Optimized Link State Routing Protocol*, IETF RFC3626, October 2003.

[14] C.Perkins, E.Belding-Royer, and S.Das, *Ad Hoc On-Demand Distance Vector(AODV) Routing*, IETF RFC3561, July 2003.

[15] IEEE, "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority", http://standards.ieee.org/db/oui/tutorials/EUI64.html, March 1997.

[16] A. Qayyum, A. Laouiti, L. Viennot, *Multipoint relaying technique for flooding broadcast messages in mobile wireless networks*, HICSS: Hawai Int. Conference on System Sciences, January 2002, Hawai, USA.

[17] C.Perkins, J.Malinen, R.Wakikawa, E.Belding-Royer, and Y.Sun, *IP Address Autoconfiguration for Ad Hoc Networks*, Internet Draft, IETF Working Group MANET, Work in progress, November 2001.

[18] K.Weniger, M.Zitterbart, *IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks*, Proceedings of European Wireless 2002, February 2002, Florence, Italy.

[19] K.Weniger, *Passive Duplicate Address Detection in Mobile Ad Hoc Networks*, Proceedings of IEEE WCNC 2003, March 2003, New Orleans, USA.

[20] K.Weniger, *Passive Autoconfiguration for Mobile Ad Hoc Networks*, Proceedings of IEEE WCNC 2003, March 2003, New Orleans, USA.

[21] N.Vaidya, *Weak Duplicate Address Detection in Mobile Ad Hoc Networks*, submitted for MobiHoc'02, June 2002.

[22] S.Nesargi, R.Prakash, *MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network*, InfoCom 2002, June 2002.

[23] Archan Misra, Subir Das, Anthony McAuley, and Sajal K.Das, *Autoconfiguration, Registration, and Mobility Management for prevasive Computing*, IEEE Personal Communication, August 2001, pp 24-31.

[24] A McAuley, S.Das, S.Baba and Y.Shobatake, *Dynamic Registration and Configuration Protocol*, draft-itsumo-drcp-00.txt, IETF, July 2000, Expired.

[25] Hongbo Zhou, Lionel M. Ni, and Matt W.Mutka, *Prophet Address Allocation for Large Scale MANETs*, IEEE INFOCOM 2003, March 2003.

[26] A. S. Tanenbaum, *Computer Networks*, Prentice Hall, 1996.

[27] Sayrafiezadeh, M., *The Birthday Problem Revisited*, Math. Mag. 67, 1994, pp 220-223.

[28] Guillaume Chelius and Eric Fleury, *Ananas : A New Adhoc Network Architectural Scheme*, INRIA Research Report 4354, January 2002. -5pt

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399